"Alexandru Ioan Cuza" University of Iasi
Faculty of Computer Science

Course
Master
I+II

Distributed Systems (I) - overview-

Lenuța Alboaie
adria@info.uaic.ro

Summary

- History & Evolution
- Distributed Systems
  - Definitions
  - Characteristics
  - Use
  - Concepts
  - Classifications
  - Necessity
- Parallel, distributed and concurrent computing

History & Evolution

- 1945-1985: “computers were large and expensive”
- … improvements:

Networking

Processors

Memory

Storage

Protocols

History & Evolution

Networking

Processors

Memory

Storage

Protocols

- Microprocessors industry (8-biti, 16,32,64,…) experienced a fast evolution
- Computers became:
  - Smaller
  - Cheaper
  - Faster
- "…from machines that cost 10 million dollars and executed 1 instruction per second we have come to machines that cost 1000 dollars and are able to execute 1 billion instructions per second, a price/performance gain of 1013"

History & Evolution

Networking

Processors

Memory

Storage

Protocols

An	Cost ($/MB)	Memory size (media)
1977	$32,000	16K
1987	$250	640K-2MB
1997	$2	64MB-256MB
2007	$0.06	512MB-2GB+
2017	$0.6($8 per GB)	8GB->...

[http://www.cs.rutgers.edu/~pxk/]

History & Evolution

Networking

Processors

Memory

Storage

Protocols

- 1977: 310KB floppy drive ~ $1480
- 1987: 40 MB drive ~ $679
- 2008: 750 GB drive ~ $99
- 2015: 1TB drive ~ $100-$120
- 2017: 4TB SATA disk drive ~$100
- *“Recording density increased over 60,000,000 times over 50 years”*

[http://www.cs.rutgers.edu/~pxk/]

History & Evolution

Networking

Processors

Memory

Storage

Protocols

1961-1972: the first communication steps in packet switching

1961: Kleinrock – has proposed a theoretical model

- 1967: ARPAnet is designed by Advanced Research Projects Agency
- 1969: appears the first operational node - ARPAnet (a network formed by 4 computers)
- 1972:
  - Public demonstration of ARPAnet underlying technologies
  - NCP (Network Control Protocol)- the first host-to-host protocol
  - The first e-mail program; the character @ is inserted for use
  - ARPAnet has 15 nodes

History & Evolution

Networking

Processors

Memory

Storage

Protocols

1972-1980: The *Internetworking* concept appears. *Proprietary networks occurs.*

- 1974: Cerf and Kahn – conceived the communication protocol - TCP (Transmission Control Protocol)
- 1976 - Robert Metcalf (Hardvard) developed the Ethernet technology which allows data transfer by coaxial cable
- 1978: TCP/IP protocol stack was standardized via RFC (Request For Comments)
- At the end of '70: proprietary architectures: DECnet, SNA, XNA
- 1979: ARPAnet had 200 nodes

History & Evolution

Networking

Processors

Memory

Storage

Protocols

LAN – the transmission speed:

- Original Ethernet: 2.94 Mbps
- **1985**: thick Ethernet: 10 Mbps; 1 Mbps with twisted pair networking
- **1991**: 10BaseT - twisted pair: 10 Mbps
- **1995**: 100 Mbps Ethernet
- **1998**: 1 Gbps (Gigabit) Ethernet
- **1999**: 802.11b (wireless Ethernet) standardized
- **2001**: 10 Gbps introduced
- **2005**: 100 Gbps (by fiber optic link)
- **2012**: …. Gbps

Large amount of data can be transferred between computers

History & Evolution

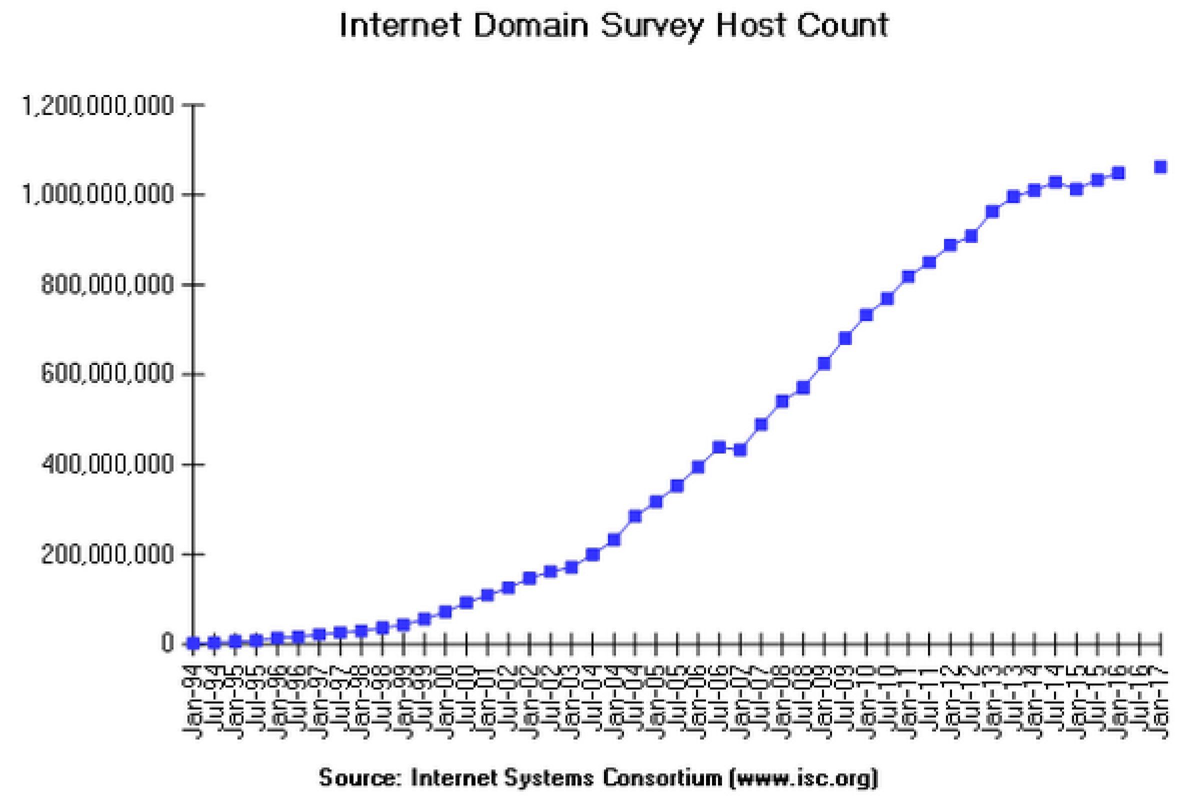


Figure. The hosts growth from January 1994 till January 2017
January 2017| Source: http://www.isc.org/solutions/survey

Distributed Systems| Definitions

- *"You know you have a distributed system when the crash of a computer you've never heard of stops you from getting any work done."*

  (**Leslie Lamport**, Distribution email, May 28, 1987, http://research.microsoft.com/users/lamport/pubs/distributed_systems.txt.)
- "*A collection of computers that do not share common memory or a common physical clock, that communicate by a messages passing over a communication network, and where each computer has its own memory and runs its own operating system. Typically the computers are semi-autonomous and are loosely coupled while they cooperate to address a problem collectively*" (**M. Singhaland N. Shivaratri**, Advanced Concepts in Operating Systems, New York, McGraw Hill, 1994)
- "*A collection of independent computers that appears to the users of the system as a single coherent computer.*" (**A. Tanenbaum** and **M. Van Steen**, Distributed Systems: Principles and Paradigms, Upper Saddle River, NJ, Prentice-Hall, 2003)

Distributed Systems| Characteristics

A distributed system - a collection of autonomous processors which communicate through a network and has the following characteristics:

- **There is no common physical clock** => asynchronies between processors
- **There is no shared memory** => m*essage-passing* mechanism for communication
  Obs. Distributed systems can supply an abstraction regarding a common address space via *distributed shared memory*
- **Geographical separation**
  - Is not required for processors to be in the same WAN
  - NOW/COW (Network/Cluster of Workstations) from a LAN are popular because of low costs and higher transfer speed
- **Autonomy and heterogeneousness**
  - The processors are *loosely coupled*
    - They have different speeds and they are use different operating systems
    - They do not belong to a dedicate system but cooperate in order to solve a problem

Distributed Systems| Characteristics

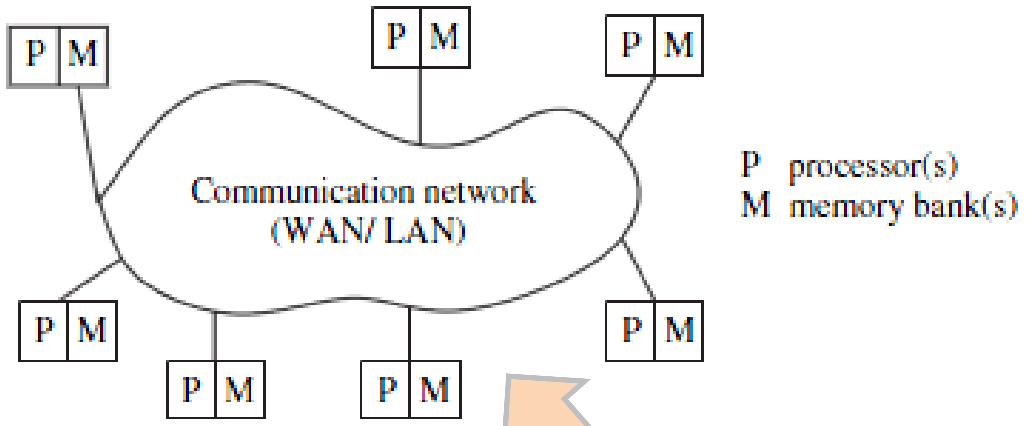


Figure. A distributed system that links nodes through a communication network

[A.Kshemkalyani , M. Singhal , Distributed Computing]

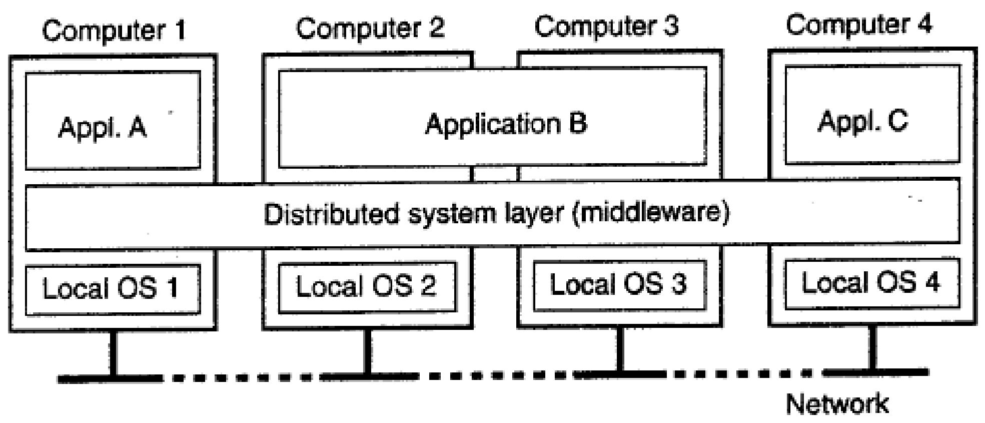
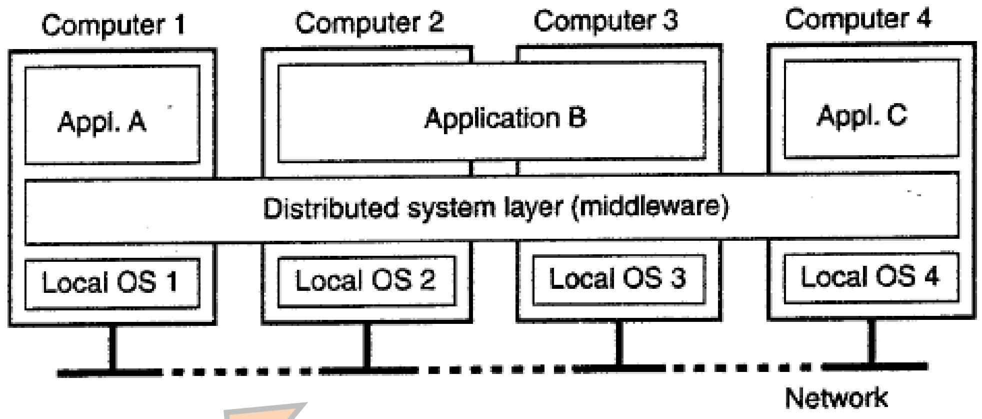


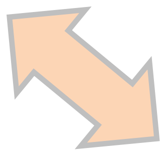
Figure. Distributed system - a generic architecture

- The levels based architecture => decreases the design complexity

[A. S. Tanenbaum, M.Steen, DISTRIBUTED SYSTEMS]

Distributed Systems| Characteristics





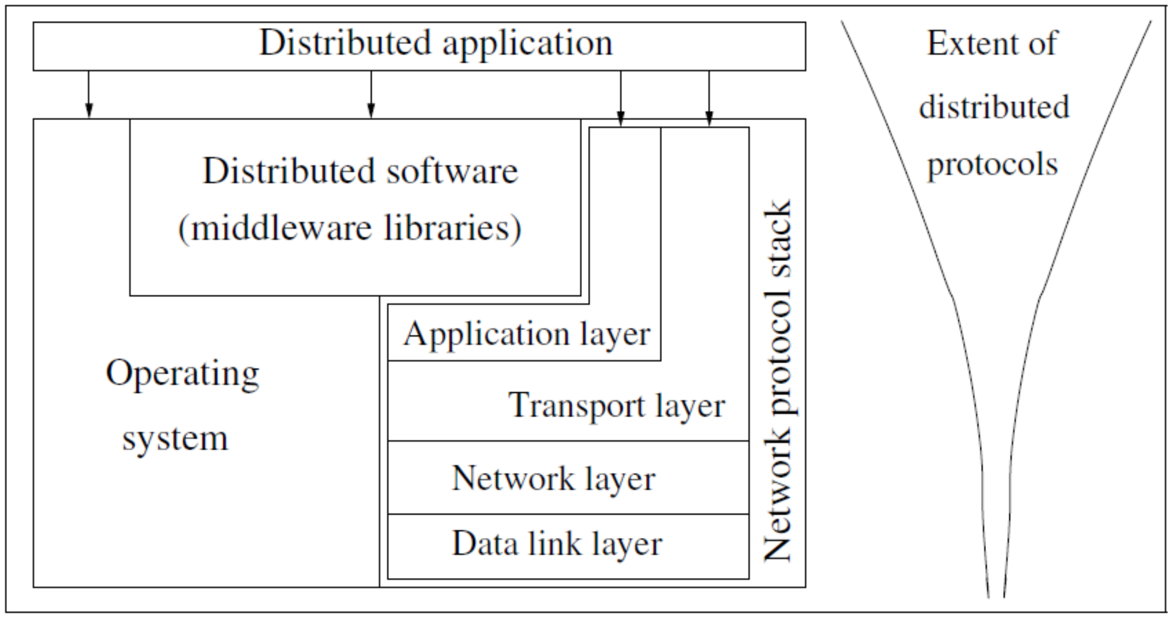


Figure. The interaction between software components

[A.Kshemkalyani , M. Singhal , Distributed Computing]

Distributed Systems| Characteristics

- Factors in the design of a distributed system:
  - Resources accessibility
  - Transparency
    - Access
    - Localization
    - Migration
    - Reallocation
    - Replication
    - Concurrency
    - Failure (e.g. *dead* versus *painfully slow*)
    - ➢ Obs. Pay attention to levels of transparency
  - *Openness*
    - Interoperability
    - Portability
  - Extensibility (Modularity and incremental expandability)

Distributed Systems| Characteristics

- Characteristics for a **well design** distributed system?
  - => “it depends”
  - …..
- Access to geographically remote data and resources
- Increased performance/cost
- Scalability
  - **size scalability**
  - **geographically scalable system**
  - **administratively scalable**
- Reliability
  - availability
  - integrity
  - fault-tolerance
  - …

Distributed Systems| Characteristics

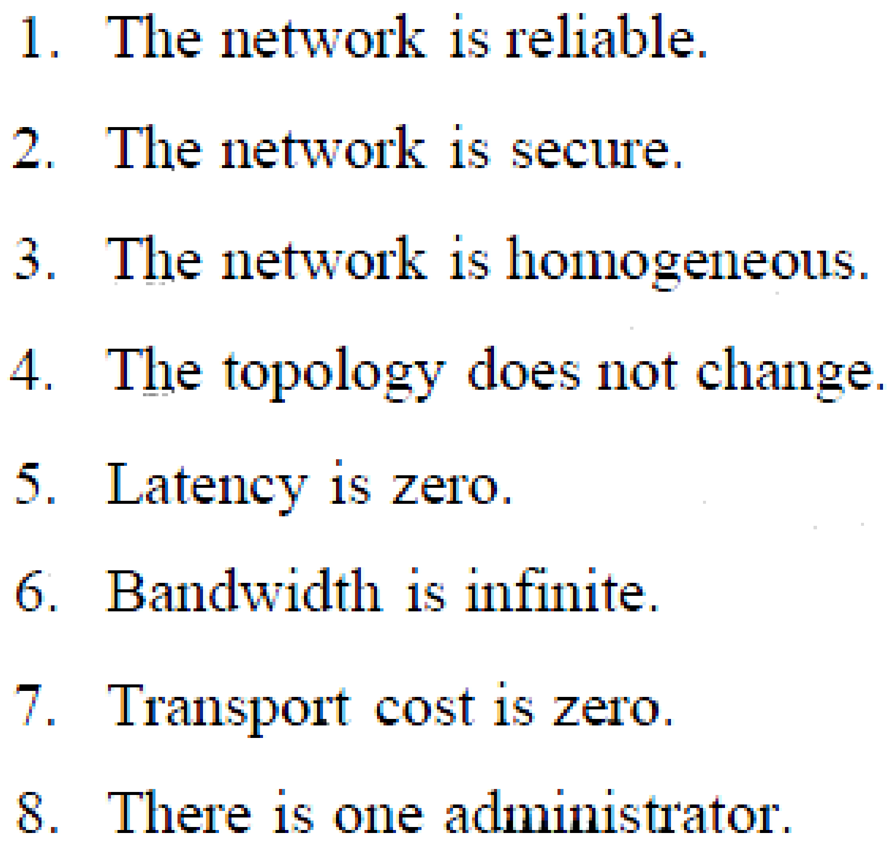
Characteristics for a **reliable** distributed system?

- Fault-Tolerant: recovering from component failures without performing incorrect actions
- Highly Available: It can restore operations, permitting it to resume providing services even when some components have failed
- Recoverable: Failed components can restart themselves and rejoin the system, after the cause of failure has been repaired.
- Consistent: The system can coordinate actions by multiple components often in the presence of concurrency and failure
- Scalable: It can operate correctly even as some aspect of the system is scaled to a larger size
- Predictable Performance: The ability to provide desired responsiveness in a timely manner.
- Secure: The system authenticates access to data and services

[Birman, Kenneth. **Reliable Distributed Systems: Technologies, Web Services and Applications.** New York: Springer-Verlag, 2005]

Distributed Systems| Characteristics

- We design a distributed system with the metaphor “design for failure” ☺
- False assumptions [Peter Deutsch, Sun Microsystems]:



1. The network is reliable.
2. The network is secure.
3. The network is homogeneous.
4. The topology does not change.
5. Latency is zero.
6. Bandwidth is infinite.
7. Transport cost is zero.
8. There is one administrator.

Distributed Systems| Failures

- Defines differences between local programming and distributed programming
- Failures
  - Hardware – dominant until 90’s; after we had to deal with hardware reliability
  - Software – determine 25-30% of system downtime
- Bugs classification for mature systems: [Gray&Reuter, ‘93]
  - Heisenbug ("Heisenberg uncertainty principle”): a bug that appears in release-mode but not under debug-mode
  - Bohrbug: a bug manifest under a well-defined conditions, and is not disappearing when it is “researched”

Distributed Systems| Failures

- “**Halting failures**: A component simply stops. There is no way to detect the failure except by timeout: it either stops sending "I'm alive" (heartbeat) messages or fails to respond to requests. Your computer freezing is a halting failure.
- **Fail-stop**: A halting failure with some kind of notification to other components. A network file server telling its clients it is about to go down is a fail-stop.
- **Omission failures**: Failure to send/receive messages primarily due to lack of buffering space, which causes a message to be discarded with no notification to either the sender or receiver. This can happen when routers become overloaded.”

[Birman, Kenneth. **Reliable Distributed Systems: Technologies, Web Services and Applications.** New York: Springer-Verlag, 2005]

Distributed Systems| Failures

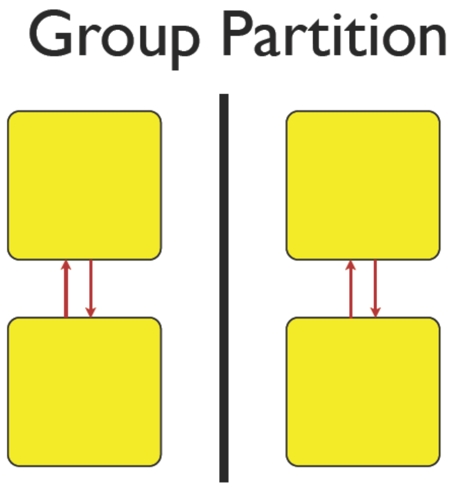
- “**Timing failures**: A temporal property of the system is violated. For example, clocks on different computers which are used to coordinate processes are not synchronized; when a message is delayed longer than a threshold period, etc.
- **Byzantine failures**: Instead of stopping, a component produces faulty data; failures caused by malicious interferences or bad hardware/software
- **Network failures**: A network link breaks.
- **Network partition failure**: A network fragments into two or more disjoint sub-networks within which messages can be sent, but between which messages are lost. This can occur due to a network failure.”

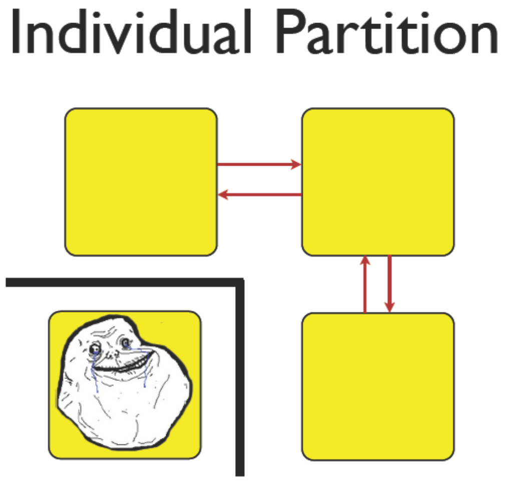
[Birman, Kenneth. **Reliable Distributed Systems: Technologies, Web Services and Applications.** New York: Springer-Verlag, 2005]

CAP |Discussions

Brewer Theorem or CAP Theorem (***Consistency***, ***Availability, Partition Tolerance)***

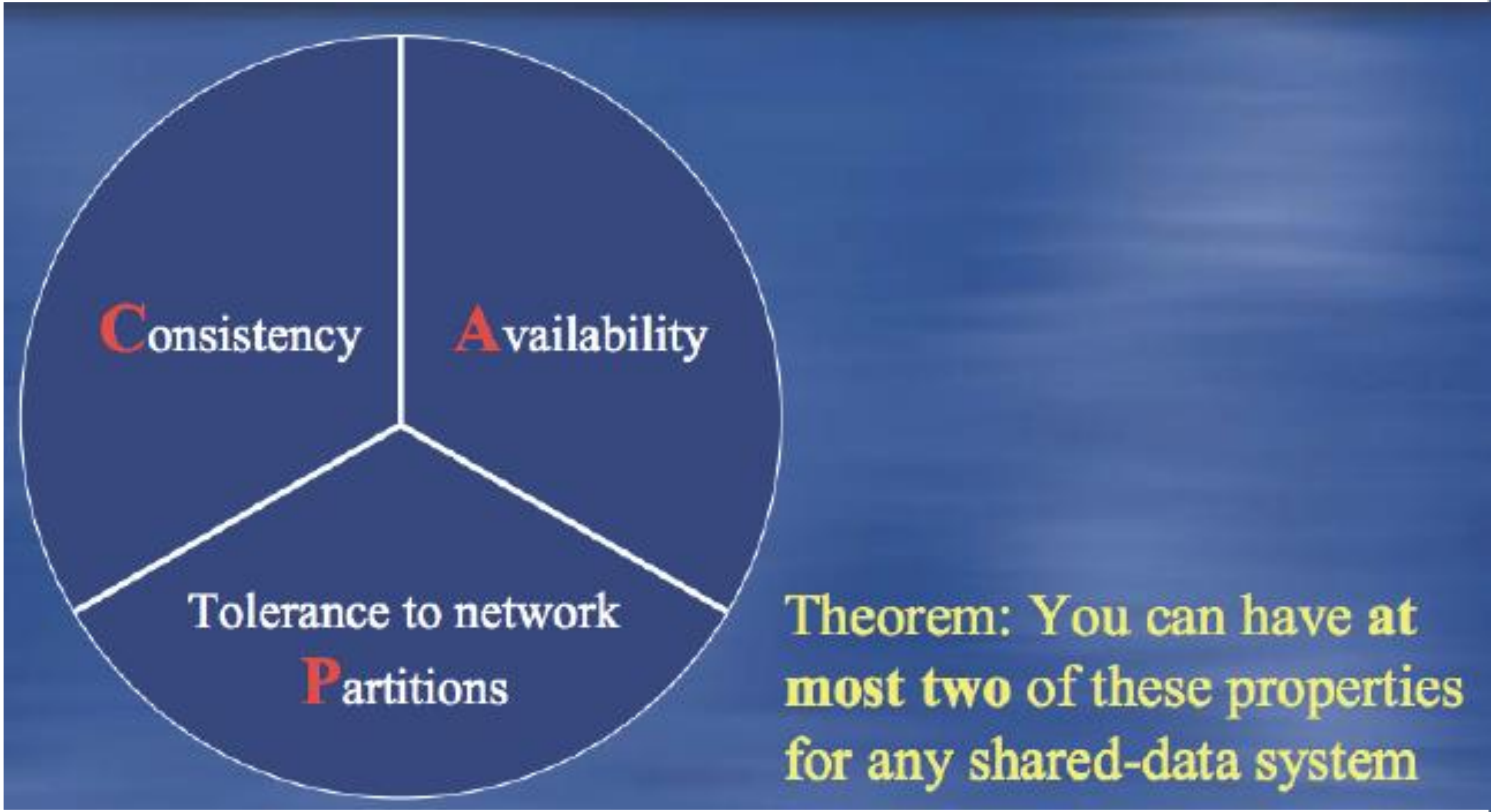
- *Consistency: atomic operation, linearizable data items* ***(C) -*** each write operation seems to happen immediately on all nodes
- *Availability* ***(A) -*** Always an answer is received for any message
- *Partition tolerance: "can lose messages (varying degree)"* ***(P)***





*Slide inspired by Ali Ghodsi, cs.berkeley.edu; Julian Browne;

CAP |Discussions



Theorem: You can have **at most two** of these properties for any shared-data system

*Slide inspired by Ali Ghodsi, cs.berkeley.edu; Julian Browne;

CAP |Discussions

- Gilbert and Lynch: formal proof of CAP theorem
- **Let's us consider an asynchronous system**
  - There isn't a global common clock
  - The agents perform local actions and communicate by messages

- **Theorem 1**: *It is impossible in the asynchronous network model to implement a read/write data object that guarantees the following properties: A (availability) and C (atomic consistency), in all fair executions (including those in which messages are lost)*
- **Argues :**
  - You can't have C, A and P if there are messages that are lost or delayed
  - How can two groups communicate updates if they can't communicate?
  - **A (**availability) requires that you return a value

*Slide inspired by Ali Ghodsi, cs.berkeley.edu; Julian Browne;

CAP |Discussions

- **Corollary 1.I**: It is impossible in the asynchronous network model to implement a read/write data object that guarantees the following properties:
  - Availability, in all fair executions
  - Atomic consistency, in fair executions in which no messages

are lost

Argues:

- In an asynchronous system is impossible to determine if a message has been delayed or if it's lost

=> In an asynchronous system we can't have C,A and P even we don't have partitions (☺ or ☹)

*Slide inspired by Ali Ghodsi, cs.berkeley.edu; Julian Browne;

CAP |Discussions

Situations:

- **C&P**
  - Never accept writes
  - Never return anything
    - => never available, so no wrong answers ☺

```
def Handle_Request(socket):
  close(socket);
  return 0;
```

- **C&A**
  - E.g. use a single master
- **A&P**
  - Always return initial value (we don't have **C**, the value is trivially available)

```
def Handle_Read(socket):
    socket.write(init_value)
    close(socket);
    return 0;
```

```
def Handle_Write(socket):
socket.write(ACK);
//do nothing
close(socket);
return 0;
```

*Slide inspired by Ali Ghodsi, cs.berkeley.edu; Julian Browne;

CAP |Discussions

- **Let us consider a partially synchronous system**
  - **With tolerance at network delays**
- **Theorem 2**: It is impossible in the partially synchronous network model to implement a read/write data object that guarantees the following properties: *Availability* and *Atomic consistency* in all executions (even those in which messages are lost).
- **Argues:**
  - In this system, if some messages are lost, writes may not be propagated correctly and you'll get stale data
  - Availability requires that you return a value

*Slide inspired by Ali Ghodsi, cs.berkeley.edu; Julian Browne;

CAP |Discussions

- ***Corollary* 2.I**: It is possible in the partially synchronous network model to implement a read/write data object that guarantees the following properties:
  - **Availability,** in all fair executions
  - **Partial atomic consistency** – in fair executions in which no messages are lost
- **Argues:**
  - In the situation in which the messages aren't los, if you don’t obtain an ACK in 2*(max_msg_transit_time)+(time_spent_processing), then there was a partition
  - Return consistent data in absence of partitions
  - Return inconsistent data with partitions and detect this is happening

*Slide inspired by Ali Ghodsi, cs.berkeley.edu; Julian Browne;

CAP | Discussions

- "Weak CAP Principle"? (HotOS 1999)
  - *"The stronger the guarantees made about any two of strong consistency, high availability, or resilience to partitions, the weaker the guarantees that can be made about the third."*
- Daniel Abadi: PACELC
  - *"if there is a partition (P) how does the system tradeoff between availability and consistency (A and C); else (E) when the system is running as normal in the absence of partitions, how does the system tradeoff between latency (L) and consistency (C)" (http://dbmsmusings.blogspot.com/2010/04/problems-with-cap-and-yahoos-little.html)*

*Slide inspired by Ali Ghodsi, cs.berkeley.edu; Julian Browne;

Distributed Design Principles

- “design for failure”
- No assumptions about components states
  - E.g. A receive a message from B, answer back assuming that B is alive….but?
- Define failure scenarios
- Client/Server must deal with unresponsive discussion partners
- Checksums for data send & received
- Minimize traffic
- Minimize latency
- Speed and performance
  - Measure and create profile for each small component
- Retransmission is costly: measures (delay retransmissions to obtain a optimal solution)

Distributed Design Principles

- Minimize stateful components
  - State: something held in one place on behalf of a process that is in another place; it cannot be reconstructed by any other component => caches are used
  - Problems: Cached data can become stale; If a process stores information that can't be reconstructed by any other component=> "Are you now a single point of failure?"
    - Solution: replication strategies
      - Problems: Partitioned network,…

Distributed System| Necessity

- Scale
- High availability
- Fault tolerance
- Reduced latency
- Delegation operations
- Collaboration
- Mobility

Distributed System| Necessity

Scale – Increase Performance

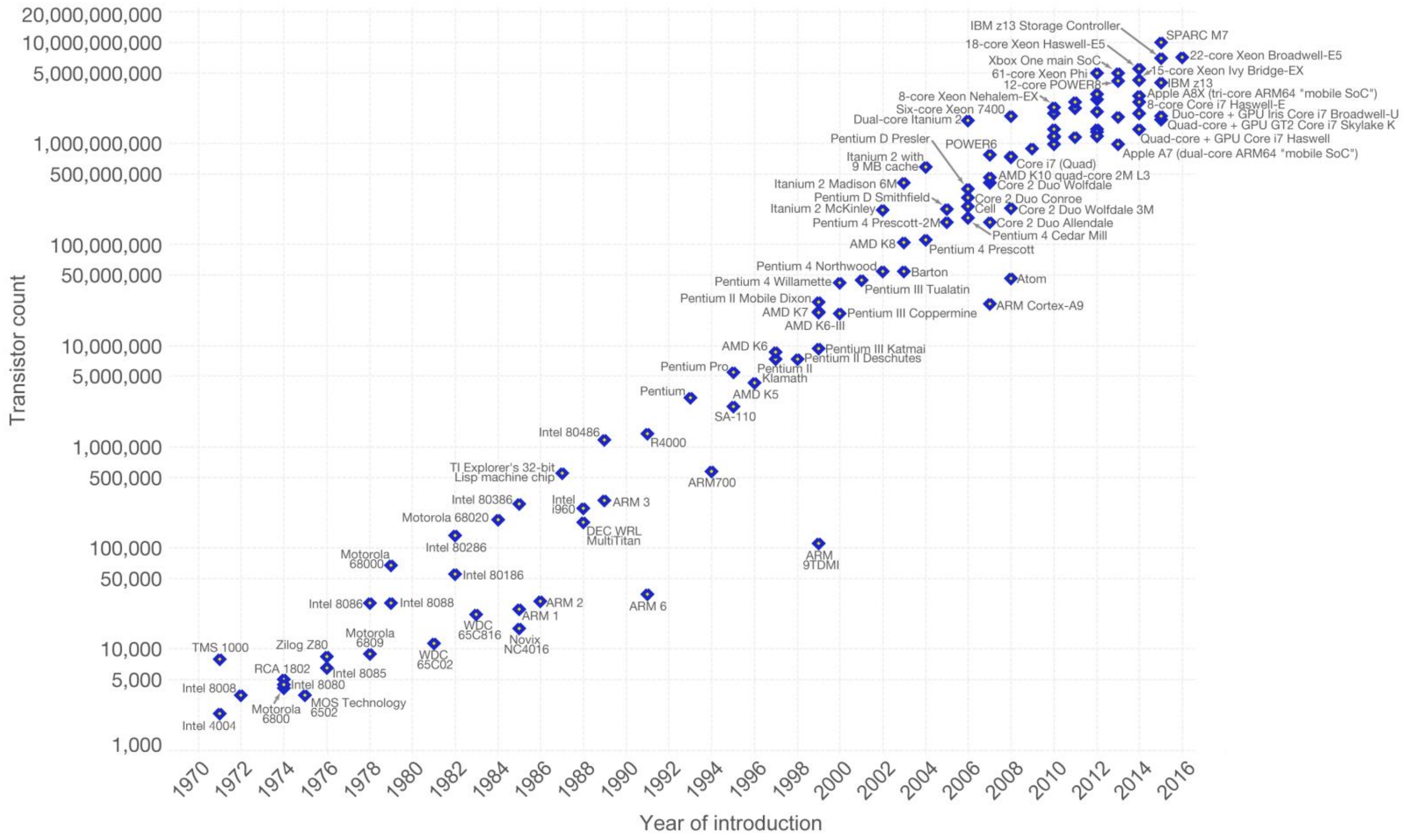
- Moore Low (1965):
  - The number of transistors in an integrated circuit doubles approximately every two years;
  - Performance double every 18 months because of faster transistors and more transistors per chip

Distributed System| Necessity

Moore's Law – The number of transistors on integrated circuit chips (1971-2016)

Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

Licensed under CC-BY-SA by the author Max Roser.

Distributed System| Necessity

Scale – Increase Performance

- Getting harder for technology to keep up with Moore's law
  - – More cores per chip → requires multithreaded programming
  - – There are limits

    Intel Broadwell Xeon CPU: 22 cores per chip (>$4,000/chip!)

    NVIDIA GeForce GTX1080: 3,584 CUDA cores per chip

    Special purpose apps: Graphics rendering, neural networks
- We want to build systems that can process billions of request per day

  Google gets over 40,000 queries per second

  Facebook: >2 billion searches per day; 13 million database queries per second

  We also want access to exabytes of storage

=>**Multiprocessor systems don't scale high**

Distributed System| Necessity

Scale – Increase Performance

=>**Multiprocessor systems don't scale high**

Example:

- "In 1999, it took Google one month to crawl and build an index of

about 50 million pages

- In 2012, the same task was accomplished in less than one minute.
- 16% to 20% of queries that get asked every day have never been asked before
- Every query has to travel on average 1,500 miles to a data center and back to return the answer to the user
- A single Google query uses 1,000 computers in 0.2 seconds to retrieve an answer"

[http://www.internetlivestats.com/google-search-statistics/]

Distributed System| Necessity

Redundancy

=>replicated components

- If p(any one system down)=5% =>

  =>P(2 systems down simultaneously)= 5%x5%=0.25%

- Systems up to provide a service: P(any system down)=100%

Availability requires

Fault tolerance

- Identify & recover from component failures

Recoverability

- May involve restoring state
- Software can restart and function
  - Problem: Stale state

[http://www.internetlivestats.com/google-search-statistics/]

Distributed System| Necessity

Reduced Latency

=> Cache data close to where it is needed (implies replication)

=> CDN (Content Delivery Network) (e.g. Akamai)

Delegated operations

- Use third-party services and someone else can manage the systems
- Dedicated systems for storage, processing et.al. (e.g. Cloud services)

Collaboration

- **Social connectivity, Commerce, News, Work & Play et.al**

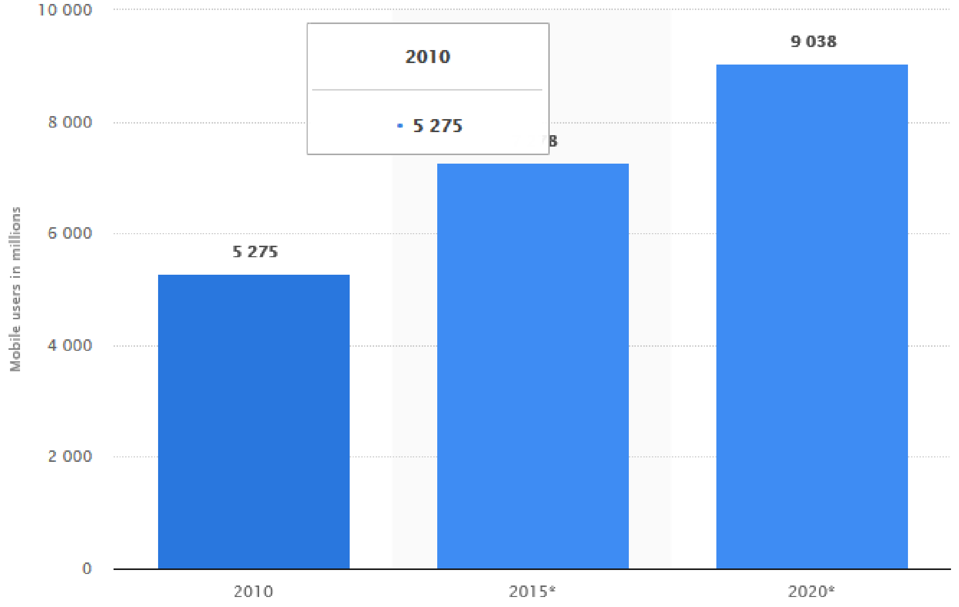
Distributed System| Necessity

Metcalfe Law

- "The value of a telecommunication network is proportional to the square of the number of connected users of the system" (George Glinder, 1993)
  - Ethernet – " compatible communicating devices – fax machines, telephones etc." (Metcalfe, 1980)

Mobility

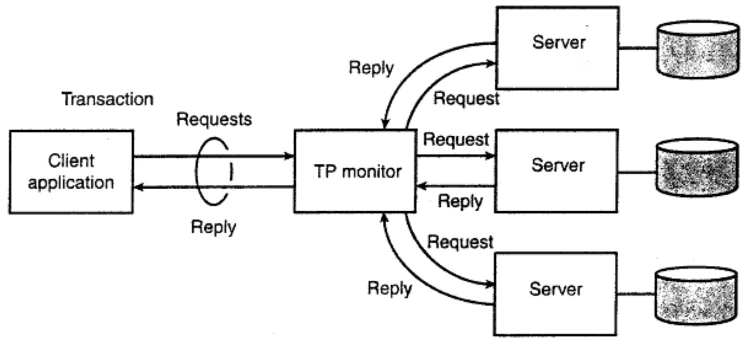
to 2020. The global number of mobile users is projected to amount to 9,038 million in 2020.



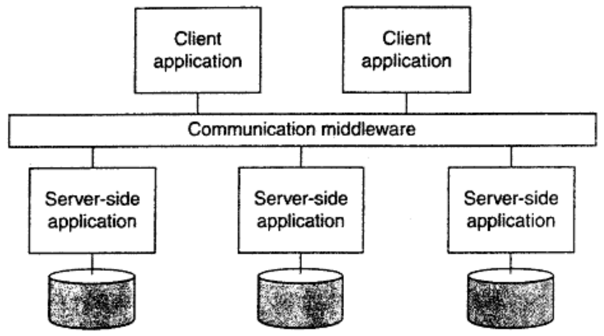
[https://www.statista.com/statistics/218984/number-of-global-mobile-users-since-2010/]

Distributed systems|Classifications

- According to [A. S. Tanenbaum, M.Steen, DISTRIBUTED SYSTEMS]:
  - *Distributed Computing Systems*
    - Systems used for realization of tasks that require high computing power
    - Examples: **Cluster Computing, Grid Computing (=> Course 3)**
  - *Distributed Information Systems*
    - Examples: Transaction Processing Systems



    - Enterprise Application Integration

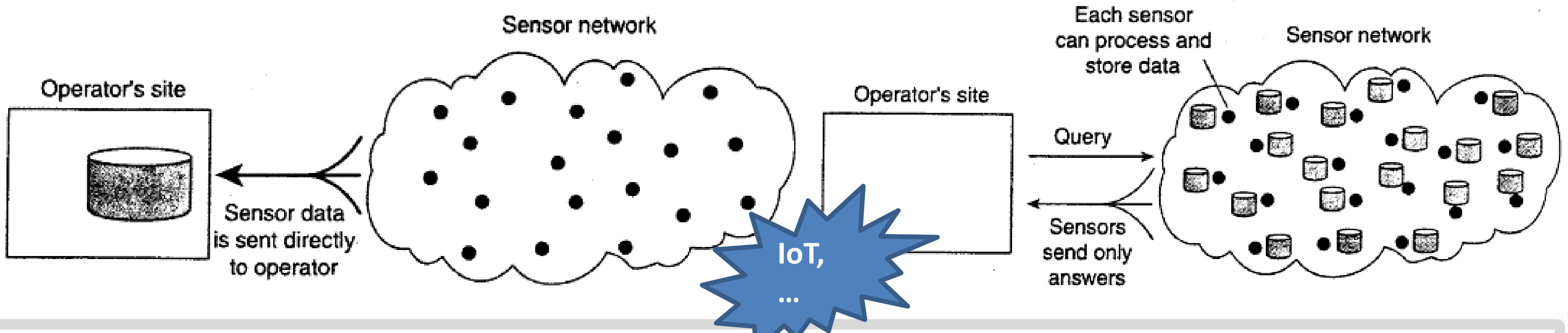


ESB/MOM,
iPaaS,
…

Middleware as a way of integration of applications (…in a future course)

Distributed systems|Classifications

- According to [A. S. Tanenbaum, M.Steen, DISTRIBUTED SYSTEMS]:
  - *Distributed Embedded Systems* or *Distributed Pervasive Systems*
    - Components:
      - Small devices – with features as: mobility, *wireless connection,* battery
    - Requirements
      - The devices are "aware" by environment changes and "nestle in" as best as possible
      - The devices has capacity to be used by different users in different ways (ad hoc composition)
      - The devices can access and also provide information
  - Exemple: *Home Systems* (smartphone, gaming devices, computers, cameras, …), *Electronic Health Care Systems,* Retele de senzori *(Sensor networks)*


Operator's site
Sensor network
Sensor data is sent directly to operator
Operator's site
Each sensor can process and store data
Sensor network
Query
Sensors send only answers
IoT, …

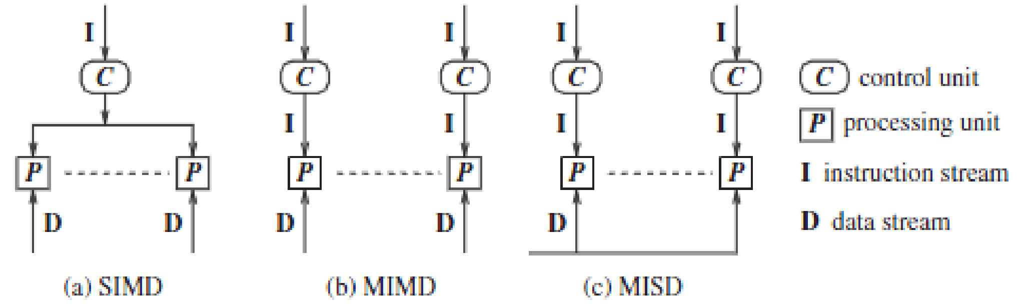
Parallel, distributed and concurrent computing

- “The terms "concurrent computing", "parallel computing", and "distributed computing" have a lot of overlap
- The processors in a typical distributed system run concurrently in parallel
  - A concurrent system is one in which we divide work into small tasks that are independent of other ongoing work and thus can be done in an interleaved or even simultaneous fashion
- Parallel computing may be seen as a particular tightly coupled form of distributed computing
- Distributed computing may be seen as a loosely coupled form of parallel computing”

Flynn taxonomy

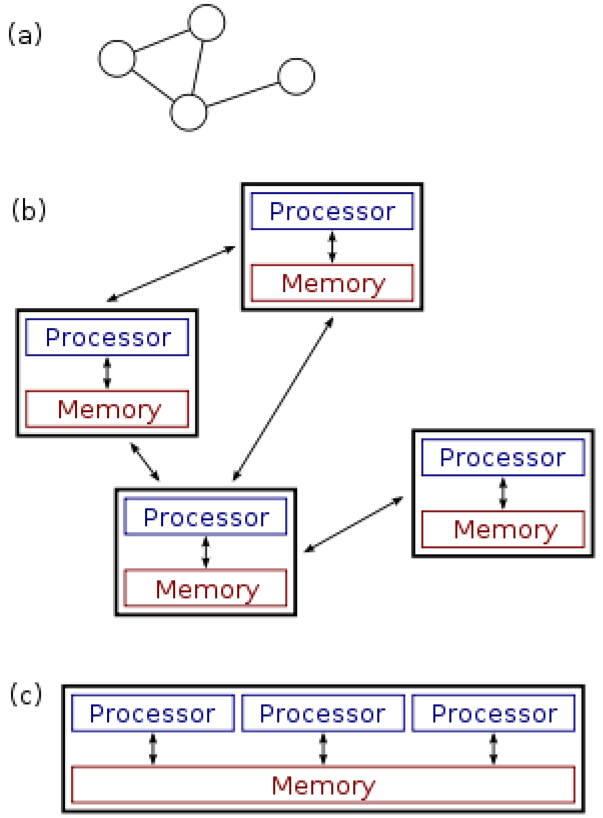
➢ Identify four processing modes

- **Single instruction stream, single data stream (SISD)**
  - Traditional uniprocessor systems
- **Single instruction stream, multiple data stream (SIMD)** *(a)*
  - multiple homogeneous processors processes different data streams
  - Ex. Applications involving operations on big matrices and vectors (scientific applications)
  - Ex. Old parallel systems: Illiac-IV, MPP, CM2, MasPar MP-1 etc.
- **Multiple instruction stream, single data stream (MISD)** *(c)*
  - Execute different operations in parallel on the same data
  - Are used in systems which assures *fault tolerance* (e.g. *Space Shuttle control system*)
- **Multiple instruction stream, multiple data stream (MIMD)** *(b)*
  - Processors execute different code on different dates; A common clock doesn't exist;
  - Many distributed systems fall into this category



Parallel, distributed and concurrent computing

- In parallel computing, all processors may have access to a shared memory to exchange information between processors
- In distributed computing, each processor has its own private memory (distributed memory). Information is exchanged by passing messages between the processors



(a), (b): a distributed system.
(c): a parallel system.

Message-passing versus shared memory

- ***Message-passing systems*** **versus** ***shared memory systems***
  - Systems with shared memory
    - There is a shared address space
    - The communication between processors is accomplished via shared variables and control variables for synchronization (semaphores, monitors)
    - For a distributed system this abstraction is called ***distributed shared memory***
  - ***message-passing (MP) mechanism*** can be simulated with ***shared memory (SM)***, and vice versa

Message-passing versus shared memory

- ***Message-passing systems*** **versus** ***shared memory systems***
  - **Emulate MP->SM**
    - Shared address space can be divided into disjoint parts and associated with each processor
    - Operations "send" and “receive" are implemented in the form of "write" and "read" and they are run by each processor in the associated space
    - A separate location is reserved as mailbox for each pair of processes
      - Example: emulate Pi-Pj ***message-passing***
        - Pi writes in *mailbox,* synchronization primitives are used for information on receiving data, Pj reads from *mailbox*
  - **Emulate SM->MP**
    - Each shared location can be modeled as a separate process
      - *write* to a shared location is emulated by sending an update message to the corresponding owner process
      - *read* to a shared location is emulated by sending a query message to the owner process

Message-passing versus shared memory

- ***Message-passing systems* versus *shared memory systems***
  - Observations:
    - Accessing another processor’s memory is expensive
    - Although emulating shared memory might seem to be more attractive from a programmer’s perspective, it must be remembered that in a distributed system, it is only an abstraction; the latencies involved in read and write operations may be high even when using shared memory emulation because the read and write operations are implemented by using network communication
    - Applications can use a combination between MP and SM (we often meet MP systems);
    - Libraries and standards: MPI (Message-Passing Interface) , PVM (parallel virtual machine) – used in scientific communities; libraries that implements RPC mechanism (Sun RPC), RMI (remote method invocation); CORBA (common object request broker architecture), DCOM (distributed component object model)

Bibliography

- Andrew S. Tanenbaum, Maarten van Steen, Distributed Systems, Principles and Paradigms, Second Edition, 2007
- Ajay D. Kshemkalyani , Mukesh Singhal , Distributed Computing **-** Principles, Algorithms, and Systems, © Cambridge University Press 2008
- Katarina Stanoevska Slabeva, Thomas Wozniak, Grid and Cloud Computing - A Business Perspective on Technology and Applications, 2010, Editors Santi Ristol, Springer-Verlag Berlin Heidelberg
- https://en.wikipedia.org/wiki/Heisenbug
- http://www.julianbrowne.com/article/viewer/brewers-cap-theorem
- Birman, Kenneth. Reliable Distributed Systems: Technologies, Web Services and Applications. New York: Springer-Verlag, 2005.
- Gray, J. and Reuter, A. Transaction Processing: Concepts and Techniques**.** San Mateo, CA: Morgan Kaufmann, 1993.

Bibliography

- Ghosh, Sukumar (2007), Distributed Systems – An Algorithmic Approach, Chapman & Hall/CRC, ISBN 978-1-58488-564-1.
- Lynch, Nancy A. (1996), Distributed Algorithms, Morgan Kaufmann, ISBN 1-55860-348-4.
- Peleg, David (2000), Distributed Computing: A Locality-Sensitive Approach, SIAM, ISBN 0-89871-464-8.
- Coulouris, George; et al. (2011), Distributed Systems: Concepts and Design (5th Edition), Addison-Wesley ISBN 0-132-14301-1.
- https://en.wikipedia.org/wiki/Distributed_computing
- https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/
- http://www.internetlivestats.com/google-search-statistics/

Summary

- History & Evolution
- Distributed Systems
  - Definitions
  - Characteristics
  - Use
  - Concepts
  - Classifications

Questions?